IN THE CLAIMS:

1. (Original) A host-fabric adapter, comprising:

at least one Micro-Engine (ME) arranged to establish connections and support data transfers, via a switched fabric, in response to work requests from a host system for data transfers;

interface blocks arranged to interface said switched fabric and said host system, and send/receive work requests and/or data for data transfers, via said switched fabric, and configured to provide context information needed for said Micro-Engine (ME) to process said work requests for data transfers, via said switched fabric,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction execution architecture to handle one or more ME instructions and/or one or more tasks so as to process data for data transfers.

2. (Currently Amended) A host-fabric adapter, comprising:

at least one Micro-Engine (ME) arranged to establish connections and support data transfers, via a switched fabric, in response to work requests from a host system for data transfers;

interface blocks arranged to interface said switched fabric and said host system, and send/receive work requests and/or data for data transfers, via said switched fabric, and configured to provide context information needed for said Micro-Engine (ME) to process said work requests for data transfers, via said switched fabric,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction execution architecture to handle one or more ME instructions and/or one or more tasks so as to process data for data transfers;

~~The host fabric adapter as claimed in claim 1,~~ wherein said Micro-Engine (ME) processes multiple ME instructions in parallel, when said ME instructions are deterministic logic and arithmetic instructions by:

processing Instruction #1 at a first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing a source address to the interface blocks for Instruction #1 at a second cycle, and processing Instruction #2 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks at a third cycle, providing the source address to the interface blocks for Instruction #2, and processing Instruction #3 in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data messages from the interface blocks for Instruction #1 at a fourth cycle, and when data for Instruction #2 is available from the interface blocks, providing the source address to the interface blocks for Instruction #3 and processing Instruction #4 in which the OpCode, source address and destination address are read from the Instruction Memory;

providing destination and write controls of Instruction #1 for the interface blocks at a fifth cycle, processing data messages from the interface blocks for Instruction #2, and when data for Instruction #3 is available from the interface blocks, providing the source address to the

interface blocks for Instruction #4 and processing Instruction #5 in which the OpCode, source address and destination address are read from the Instruction Memory; and

when Instruction #1 is retired at a sixth cycle, providing destination and write controls of Instruction #2 for the interface blocks, processing the data from the interface blocks for Instruction #3, and when data for Instruction #4 is available from the interface blocks, providing the source address to the interface blocks for the instruction #5 and processing Instruction #6 in which the OpCode, source address and destination address are read from the Instruction Memory.

3. (Original) The host-fabric adapter as claimed in claim 2, wherein said Micro-Engine (ME) is configured to ensure that only latest data from the interface blocks is used and correct data is written to the interface blocks.

4. (Currently Amended) A host-fabric adapter, comprising:

at least one Micro-Engine (ME) arranged to establish connections and support data transfers, via a switched fabric, in response to work requests from a host system for data transfers;

interface blocks arranged to interface said switched fabric and said host system, and send/receive work requests and/or data for data transfers, via said switched fabric, and configured to provide context information needed for said Micro-Engine (ME) to process said work requests for data transfers, via said switched fabric,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction execution architecture to handle one or more ME instructions and/or one or more tasks so as to process data for data transfers;

~~The host fabric adapter as claimed in claim 1,~~ wherein said Micro-Engine (ME) processes multiple ME instructions in parallel, when said ME instructions are non-deterministic logic and arithmetic instructions by:

processing Instruction #1 at a first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing the source address to the interface blocks for Instruction #1 at a second cycle, and processing Instruction #2 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks at a third cycle, and a conditional Jump instruction based on Flags is set for Instruction #1, processing Instruction #3 in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data from the interface blocks for Instruction #1 at a fourth cycle, providing the source address to the interface blocks for Instruction #3, processing Instruction #4 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #3 is available from the interface blocks at a fifth cycle, providing destination and write controls of Instruction #4 for the interface blocks;

if the Jump condition is not TRUE, processing Instruction #5 in which the OpCode, source address and destination address are read from the Instruction Memory;

if the Jump condition is TRUE, processing the conditional Jump instruction in which the OpCode, source address and destination address are read from the Instruction Memory corresponding to a Jump Address;

when Instruction #1 is retired at a sixth cycle, flushing Instruction #3 and data for Instruction #4 available from the interface blocks, and providing the source address to the interface blocks for the conditional Jump instruction corresponding to the Jump Address if the Jump condition is TRUE; and

if the Jump condition is FALSE, providing the source address to the interface blocks for Instruction #5 and processing the conditional Jump instruction in which the OpCode, source address and destination address are read from the Instruction Memory corresponding to the Jump Address.

5. (Original) The host-fabric adapter as claimed in claim 4, wherein said Micro-Engine (ME) is configured to ensure that only latest data from the interface blocks is used and correct data is written to the interface blocks.

6. (Currently Amended) A host-fabric adapter, comprising:

at least one Micro-Engine (ME) arranged to establish connections and support data transfers, via a switched fabric, in response to work requests from a host system for data transfers;

interface blocks arranged to interface said switched fabric and said host system, and send/receive work requests and/or data for data transfers, via said switched fabric, and

configured to provide context information needed for said Micro-Engine (ME) to process said work requests for data transfers, via said switched fabric,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction execution architecture to handle one or more ME instructions and/or one or more tasks so as to process data for data transfers;

~~The host-fabric adapter as claimed in claim 1,~~ wherein said Micro-Engine (ME) processes multiple tasks in parallel by:

processing Instruction #1 at a first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing the source address to the interface blocks for Instruction #1 at a second cycle, and processing Instruction #2 indicating a Task Switching Instruction in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks and there is no data processing at a third cycle, processing Instruction #3 for a new task in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data for Instruction #1 from the interface blocks at a fourth cycle and providing the source address to the interface blocks for Instruction #3 for the new task;

providing destination and write controls of Instruction #1 for the interface blocks at a fifth cycle and, when data for the new task for Instruction #3 is available from the interface blocks, providing the source address to the interface blocks for Instruction #4 and processing Instruction #5 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory;

when Instruction #1 is retired at a sixth cycle, processing data from the interface blocks for Instruction #3 for the new task, and when data for the new task for Instruction #4 is available from the interface blocks, providing the source address to the interface blocks for Instruction #5 and processing Instruction #6 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory; and

when Instruction #2 is retired at a seventh cycle, providing destination and write controls for the interface blocks for Instruction #3, processing data from the interface blocks for Instruction #4 for the new task, and when data for the new task for Instruction #5 is available from the interface blocks, providing the source address to the interface blocks for Instruction #6 and processing Instruction #7 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory.

7. (Original) The host-fabric adapter as claimed in claim 6, wherein said Micro-Engine (ME) is configured to ensure that only latest data from the interface blocks is used and correct data is written to the interface blocks.

8. (Original) The host-fabric adapter as claimed in claim 1, wherein said interface blocks comprises:

a serial interface arranged to receive and transmit data from said switched fabric for data transfers;

a host interface arranged to receive and transmit work requests, in the form of work queue elements (WQEs), from said host system for data transfers;

a context memory arranged to store context information needed for said Micro-Engine (ME) to process work requests for data transfers;

a first-in/first-out (FIFO) interface arranged to receive data from said switched fabric via said serial interface, and to transmit data to said switched fabric via said serial interface;

an address translation interface arranged for address translation from said Micro-Engine (ME);

a local bus interface arranged to support system accessible context connections and data transfers; and

a completion queue/doorbell manager interface arranged to provide an interface to completion queues, and to update the context information needed for said Micro-Engine (ME) to process work requests for data transfers.

9. (Currently Amended) The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine (ME) comprises:

one or more Data Multiplexers arranged to supply appropriate interface data based on an ME instruction;

an Instruction Memory arranged to provide said ME instruction based on downloadable MicroCode;

an Arithmetic Logic Unit (ALU) arranged to perform mathematical, logical and shifting operations, and supply write data to ~~the~~ a host interface, ~~the~~ an address translation interface, ~~the~~ a context memory interface, ~~the~~ a local bus interface, ~~the~~ a completion queue/doorbell manager interface, and ~~the~~ a FIFO interface, via a system data bus; and

an Instruction Decoder arranged to supply system controls to the host interface, the address translation interface, the context memory interface, the local bus interface, the completion queue/doorbell manager interface, and the FIFO interface, via a system control bus, to execute said ME instruction from said Instruction Memory to control operations of said Data Multiplexers, and to determine functions of said Arithmetic Logic Unit (ALU).

10. (Original) The host-fabric adapter as claimed in claim 9, wherein said Instruction Memory corresponds to a random-access-memory (RAM) provided to store MicroCode that are downloadable for providing said ME instruction to said Instruction Decoder.

11. (Currently Amended) The host-fabric adapter as claimed in claim 10, wherein said Micro-Engine (ME) and said interface blocks are configured ~~in accordance~~ to be compliant with the *"InfiniBand™ Specification"*, and are implemented as part of an Application Specific Integrated Circuit (ASIC).

12. (Original) A host-fabric adapter installed at a host system for connecting to a switched fabric of a data network, comprising:

at least one Micro-Engine (ME) arranged to establish connections and support data transfers via said switched fabric;

a serial interface arranged to receive and transmit data from said switched fabric for data transfers;

a host interface arranged to receive and transmit work requests from said host system for data transfers; and

a context memory interface arranged to store context information needed for said Micro-Engine (ME) to process work requests for data transfers,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction execution architecture to handle one or more ME instructions and/or one or more tasks in parallel so as to process data for data transfers.

13. (Currently Amended) A host-fabric adapter installed at a host system for connecting to a switched fabric of a data network, comprising:

at least one Micro-Engine (ME) arranged to establish connections and support data transfers via said switched fabric;

a serial interface arranged to receive and transmit data from said switched fabric for data transfers;

a host interface arranged to receive and transmit work requests from said host system for data transfers; and

a context memory interface arranged to store context information needed for said Micro-Engine (ME) to process work requests for data transfers,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction execution architecture to handle one or more ME instructions and/or one or more tasks in parallel so as to process data for data transfers;

~~The host-fabric adapter as claimed in claim 12,~~ wherein said Micro-Engine (ME) processes multiple ME instructions in parallel, when said ME instructions are deterministic logic and arithmetic instructions by:

processing Instruction #1 at a first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing a source address to the interface blocks for Instruction #1 at a second cycle, and processing Instruction #2 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks at a third cycle, providing the source address to the interface blocks for Instruction #2, and processing Instruction #3 in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data messages from the interface blocks for Instruction #1 at a fourth cycle, and when data for Instruction #2 is available from the interface blocks, providing the source address to the interface blocks for Instruction #3 and processing Instruction #4 in which the OpCode, source address and destination address are read from the Instruction Memory;

providing destination and write controls of Instruction #1 for the interface blocks at a fifth cycle, processing data messages from the interface blocks for Instruction #2, and when data for Instruction #3 is available from the interface blocks, providing the source address to the interface blocks for Instruction #4 and processing Instruction #5 in which the OpCode, source address and destination address are read from the Instruction Memory; and

when Instruction #1 is retired at a sixth cycle, providing destination and write controls of Instruction #2 for the interface blocks, processing the data from the interface blocks for Instruction #3, and when data for Instruction #4 is available from the interface blocks, providing the source address to the interface blocks for the instruction #5 and processing Instruction #6 in

which the OpCode, source address and destination address are read from the Instruction Memory.

14. (Original) The host-fabric adapter as claimed in claim 13, wherein said Micro-Engine (ME) is configured to ensure that only latest data from the interface blocks is used and correct data is written to the interface blocks.

15. (Currently Amended) A host-fabric adapter installed at a host system for connecting to a switched fabric of a data network, comprising:

at least one Micro-Engine (ME) arranged to establish connections and support data transfers via said switched fabric;

a serial interface arranged to receive and transmit data from said switched fabric for data transfers;

a host interface arranged to receive and transmit work requests from said host system for data transfers; and

a context memory interface arranged to store context information needed for said Micro-Engine (ME) to process work requests for data transfers,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction execution architecture to handle one or more ME instructions and/or one or more tasks in parallel so as to process data for data transfers;

~~The host-fabric adapter as claimed in claim 12,~~ wherein said Micro-Engine (ME) processes multiple ME instructions in parallel, when said ME instructions are non-deterministic logic and arithmetic instructions by:

processing Instruction #1 at a first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing the source address to the interface blocks for Instruction #1 at a second cycle, and processing Instruction #2 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks at a third cycle, and a conditional Jump instruction based on Flags is set for Instruction #1, processing Instruction #3 in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data from the interface blocks for Instruction #1 at a fourth cycle, providing the source address to the interface blocks for Instruction #3, processing Instruction #4 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #3 is available from the interface blocks at a fifth cycle, providing destination and write controls of Instruction #4 for the interface blocks;

if the Jump condition is not TRUE, processing Instruction #5 in which the OpCode, source address and destination address are read from the Instruction Memory;

if the Jump condition is TRUE, processing the conditional Jump instruction in which the OpCode, source address and destination address are read from the Instruction Memory corresponding to a Jump Address;

when Instruction #1 is retired at a sixth cycle, flushing Instruction #3 and data for Instruction #4 available from the interface blocks, and providing the source address to the interface blocks for the conditional Jump instruction corresponding to the Jump Address if the Jump condition is TRUE; and

if the Jump condition is FALSE, providing the source address to the interface blocks for Instruction #5 and processing the conditional Jump instruction in which the OpCode, source address and destination address are read from the Instruction Memory corresponding to the Jump Address.

16. (Original) The host-fabric adapter as claimed in claim 15, wherein said Micro-Engine (ME) is configured to ensure that only latest data from the interface blocks is used and correct data is written to the interface blocks.

17. (Currently Amended) A host-fabric adapter installed at a host system for connecting to a switched fabric of a data network, comprising:

at least one Micro-Engine (ME) arranged to establish connections and support data transfers via said switched fabric;

a serial interface arranged to receive and transmit data from said switched fabric for data transfers;

a host interface arranged to receive and transmit work requests from said host system for data transfers; and

a context memory interface arranged to store context information needed for said Micro-Engine (ME) to process work requests for data transfers,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction execution architecture to handle one or more ME instructions and/or one or more tasks in parallel so as to process data for data transfers;

~~The host fabric adapter as claimed in claim 12,~~ wherein said Micro-Engine (ME) processes multiple tasks in parallel by:

processing Instruction #1 at a first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing the source address to the interface blocks for Instruction #1 at a second cycle, and processing Instruction #2 indicating a Task Switching Instruction in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks and there is no data processing at a third cycle, processing Instruction #3 for a new task in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data for Instruction #1 from the interface blocks at a fourth cycle and providing the source address to the interface blocks for Instruction #3 for the new task;

providing destination and write controls of Instruction #1 for the interface blocks at a fifth cycle and, when data for the new task for Instruction #3 is available from the interface blocks, providing the source address to the interface blocks for Instruction #4 and processing Instruction #5 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory;

when Instruction #1 is retired at a sixth cycle, processing data from the interface blocks for Instruction #3 for the new task, and when data for the new task for Instruction #4 is available from the interface blocks, providing the source address to the interface blocks for Instruction #5 and processing Instruction #6 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory; and

when Instruction #2 is retired at a seventh cycle, providing destination and write controls for the interface blocks for Instruction #3, processing data from the interface blocks for Instruction #4 for the new task, and when data for the new task for Instruction #5 is available from the interface blocks, providing the source address to the interface blocks for Instruction #6 and processing Instruction #7 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory.

18. (Original) The host-fabric adapter as claimed in claim 17, wherein said Micro-Engine (ME) is configured to ensure that only latest data from the interface blocks is used and correct data is written to the interface blocks.

19. (Original) The host-fabric adapter as claimed in claim 12, wherein said Micro-Engine (ME) comprises:

one or more Data Multiplexers arranged to supply appropriate interface data based on an ME instruction;

an Instruction Memory arranged to provide said ME instruction based on downloadable MicroCode;

an Arithmetic Logic Unit (ALU) arranged to perform mathematical, logical and shifting operations, and supply write data to the host interface, the address translation interface, the context memory interface, the local bus interface, the completion queue/doorbell manager interface, and the FIFO interface, via a system data bus; and

an Instruction Decoder arranged to supply system controls to the host interface, the address translation interface, the context memory interface, the local bus interface, the completion queue/doorbell manager interface, and the FIFO interface, via a system control bus, to execute said ME instruction from said Instruction Memory to control operations of said Data Multiplexers, and to determine functions of said Arithmetic Logic Unit (ALU).

20. (Currently Amended) The host-fabric adapter as claimed in claim 19, wherein said Micro-Engine (ME) and said appropriate interface are configured ~~in accordance~~ to be compliant with the *"InfiniBand™ Specification"*, and are implemented as part of an Application Specific Integrated Circuit (ASIC).

Please add new claims 21-26 as follows:

21. (New) The host-fabric adapter as claimed in claim 8, wherein said Micro-Engine (ME) comprises:

one or more Data Multiplexers arranged to supply appropriate interface data based on an ME instruction;

an Instruction Memory arranged to provide said ME instruction based on downloadable MicroCode;

an Arithmetic Logic Unit (ALU) arranged to perform mathematical, logical and shifting operations, and supply write data to the host interface, the address translation interface, the

context memory interface, the local bus interface, the completion queue/doorbell manager interface, and the FIFO interface, via a system data bus; and

an Instruction Decoder arranged to supply system controls to the host interface, the address translation interface, the context memory interface, the local bus interface, the completion queue/doorbell manager interface, and the FIFO interface, via a system control bus, to execute said ME instruction from said Instruction Memory to control operations of said Data Multiplexers, and to determine functions of said Arithmetic Logic Unit (ALU).

22. (New) The host-fabric adapter as claimed in claim 21, wherein the Instruction Memory corresponds to a random-access-memory (RAM) provided to store MicroCode that are downloadable for providing the ME instruction to the Instruction Decoder.

23. (New) The host-fabric adapter as claimed in claim 22, wherein the Micro-Engine (ME) and the appropriate interface are configured to be compliant with the *"InfiniBand™ Specification"*, and are implemented as part of an Application Specific Integrated Circuit (ASIC).

24. (New) The host-fabric adapter as claimed in claim 18, wherein said Micro-Engine (ME) comprises:

one or more Data Multiplexers arranged to supply appropriate interface data based on an ME instruction;

an Instruction Memory arranged to provide said ME instruction based on downloadable MicroCode;

an Arithmetic Logic Unit (ALU) arranged to perform mathematical, logical and shifting operations, and supply write data to the host interface, the address translation interface, the

context memory interface, the local bus interface, the completion queue/doorbell manager interface, and the FIFO interface, via a system data bus; and

an Instruction Decoder arranged to supply system controls to the host interface, the address translation interface, the context memory interface, the local bus interface, the completion queue/doorbell manager interface, and the FIFO interface, via a system control bus, to execute said ME instruction from said Instruction Memory to control operations of said Data Multiplexers, and to determine functions of said Arithmetic Logic Unit (ALU).

25. (New) The host-fabric adapter as claimed in claim 24, wherein the Instruction Memory corresponds to a random-access-memory (RAM) provided to store MicroCode that are downloadable for providing the ME instruction to the Instruction Decoder.

26. (New) The host-fabric adapter as claimed in claim 25, wherein the Micro-Engine (ME) and the appropriate interface are configured to be compliant with the *"InfiniBand™ Specification"*, and are implemented as part of an Application Specific Integrated Circuit (ASIC).